Reference 3
Japanese Patent Application Public-disclosure No. 1-100638
Japanese Patent Application Public-disclosure date: April 18, 1989
Title of the invention: Instruction retry control system
Japanese Patent Application No. 62-259341
Japanese Patent Application date: October 14, 1987

[Embodiment]

Next, an embodiment of the present invention will be described with reference to the attached drawings.

Referring to Fig. 1, the embodiment of the present invention comprises: operation control means 1 for controlling operation execution (EX) after each processing indicated in Fig. 4, i.e., instruction fetch (IF), operand address creation (AC), address translation (AC) and operand cache access (CA); software visible register 2; first update instruction register PCC 3 for storing an update instruction signal for the register 2 provided from the operation control means 1 via the line 101; first type indication register RID 4 for storing a type indication signal for the register 2 provided from the operation control means 1 via the line 102; first number register RND 5 for storing a register number of the register 2 provided from the operation control means 1 via the line 103; updated data register RDR 6 for storing updated data for the register 2 provided from the operation control means 1 via the line 104; second update instruction register PCC' 7 for storing an update instruction signal from the first update instruction register PCC 3; RID' 8 for storing a type indication signal from the first type indication register RID 4; second number register RNO' 9 for storing a register number from the first number register RNO 5; read data register RDR' 10 for storing pre-write data from the software visible register 2; register save buffer save/recovery control means 11 for instructing to write the pre-update data to the register save buffer 13 in response to an update of the software visible register 2 and to write all

the valid data in the register save buffer 13 back to the software visible register 2 in response to recovery; register save buffer 13 for storing pre-update data in response to an instruction from the control means 11 and reading the valid data; register save buffer clear control means 21 for instructing the register save buffer 13 to conduct the clear procedure; save address register SBA 12 for providing a save address to the register save buffer 13 via the line 115; counter CNT 14 for updating the content of the register SBA 12; instruction counter 17; CNT 18 for updating the content of the instruction counter 17; failure processing control means 16 for controlling failure processing, sending out a failure detection signal via the line 108 and sending out a recovery instruction signal to the save/recovery control means 11 via the line 111; instruction counter update control means 15 for controlling an update of the content of the instruction counter 17 in response to a failure detection signal provided via the line 108; register save buffer counter 20 for holding a value representing the number of contents of the software visible register 2 updated by a specific instruction; and register buffer counter control means 19 for, as the order of an update of software visible register 2 does not agree with the serial order of instructions before they enter the operation means due to multiple pipelines of different stages, recognizing and controlling which instruction updates how many values in the software visible register 2 and which counter (I), (II) or (III) in the register save buffer counter 20 holds the counter number representing the updated value(s).

Referring to Fig. 2, the operation control means 1 in Fig. 1 consists of pipelines divided into five stages and comprises: registers 31 ~ 36 for storing at each stage of a pipeline an update instruction signal for the software visible register 2 as a floating-point arithmetic operation is performed; registers 37 ~ 42 for storing at each stage of the pipeline a type indication signal for the software visible register 2 to be updated as a floating-point arithmetic operation is

performed; registers 43 ~ 48 for storing at each stage of the pipeline a register number of software visible register 2 to be updated as a floating-point arithmetic operation is performed; registers 49 ~ 54 for performing a floating-point arithmetic operation; shift circuits 55 and 59; operators 56 ~ 58; registers 60 and 61 for storing an update instruction signal for the software visible register 2 only at the first stage of the pipeline as a fixed-point arithmetic operation is performed; registers 62 and 63 for storing only at the first stage of the pipeline a type indication signal for the software visible register 2 to be updated as a fixed-point arithmetic operation is performed; registers 64 and 65 for storing only at the first stage of the pipeline a register number of the software visible register 2 to be updated as a fixed-point arithmetic operation is performed; registers 66 and 67 and operator 68 for performing fixed-point arithmetic operations; selector 69 for selecting an update instruction signal either from the floating point register 36 or the fixed-point register 61 and sending the selected signal to the line 101; selector 70 for selecting a type indication signal either from the floating-point register 42 or the fixed-point register 63 and sending out the thus selected signal to the line 102; selector 71 for selecting a register number either from the floating point register 48 or the fixed point register 65; and selector 72 for selecting an operation result either of the floating point register 54 or the fixed point register 67 and sending out the selected operation result.

First, an operation on which the embodiment of the present invention is based will be explained in detail. It is first assumed here that the instruction A requires long pipeline operation processing such as a floating-point arithmetic instruction.

Referring to Fig. 3, the instruction A is fetched by performing the process at the instruction fetch (IF) stage in the cycle 1. Next, in the cycle 2, an address for the instruction A is generated by performing the process at the

address creation (AC) stage, while an instruction B which only requires a short pipeline operation such as a fixed point arithmetic instruction is fetched by performing the process at the instruction fetch (IF) stage. Subsequently, address translation (AT) process and cache access (CA) process are performed on the instruction A in a similar manner and a floating-point arithmetic operation begins in the cycle 5. After address creation (AC) process, address translation (AC) process and cache access (CA) process are performed on the instruction B, an arithmetic operation begins in the cycle 6, which operation is performed by 16 bits of the 32-bit fixed point data.

After instruction fetch (IC) process, address creation (AC) process and cache access (CA) process are conducted on the instruction C following the instruction B, an arithmetic operation is performed by 1 byte of the 1-byte fixed point data.

As can be seen from Figs. 1, 2 and 3, an arithmetic operation on the instruction A is conducted in such a manner that after shift process is performed by the shift circuit 55 in the cycle 5, the operators 56, 57 and 58 perform floating-point arithmetic operations in the cycles 6, 7 and 8 respectively.

The operand of the instruction B is fetched in the cycle 5 and in the next cycle 6, the fixed point operator 68 performs an arithmetic operation on the first half of the data, i. e., 16 bits of the data. Then, in the cycle 7, the operator 68 performs an arithmetic operation on the latter half of the data, that is, the remaining 16 bits of the data. The result of the arithmetic operation is set in the data register 6 via the register 54 and selector 72.

Next, a read/write operation from the software visible register 2 will be explained in detail.

As can be seen from Figs. 1 and 3, the result of the operation is set in the updated data register 6 in the cycle 7 and an update instruction signal is set in the update instruction register PCC 3, and further, a type indication

signal is set in the first type indication register RID 4 and a register number is set in the first number register RNO 5. The update instruction signal is provided to the register buffer save/recovery control means 11 via the register 7 and line 105 to initialize the save buffer address register 12 via the line 106. An address in the register 12 is counted up by the counter CNT 14 by +1 every cycle. Therefore, in the next cycle 8, pre-update data B0' is read from the software visible register 2 based on the type indication signal from the register 4 and register number from the register RNO 5 in response to the update instruction signal from the register 3. The thus read data is stored in the register save buffer 13 via the register 10.

Assuming that a validity bit is "1" at this time, a write instruction signal is provided from the save/recovery control means 11 via the line 116, while the address "1" is provided from the address register 12 via the line 115. At this time, the type indication signal from the register 8 and the register number from the register 9 are stored in the register save buffer 13.

On the other hand, the result of the operation from the register 6 is stored in the software visible register 2. In the cycle 8, the register save buffer counter control means 19 is driven from the save/recover control means 11 via the line 118 and as a result, the instruction B is set in the pseudo instruction counter (IC') in the control means 19, which controls the corresponding counter (II) of the register buffer counter 20 initialized via the signal line 117 to count the instruction B.

The instruction counter update control means 15 is driven from the operation control means 1 via the line 107. The instruction counter update control means 15 determines by means of a signal sent via the line 108 that there is no failure detected, further determines, by means of a signal sent via the line 107, that the instruction B being executed at the operation control means 1 is not the originally expected instruction A and counts the instruction B not in the instruction counter 17 but in the pseudo

instruction counter in the register save buffer counter control means 19.

In the cycle 9, the same operation as in the cycle 8 is performed on the next instruction or data.

One of the characteristics of the present invention lies in the instruction retry executed when a failure is detected during execution of an arithmetic operation at the stage EX5. In other words, the failure processing control means 16 notifies the instruction counter update control means 15 via the line 108 of the failure detection. In the cycle 10, the control means 15 resets the pseudo-instruction counter in the control means 19 and sets the instruction A in the instruction counter 17 via the line 109.

In the cycle 10, the pre-update data C' is saved from the software visible register 2 in the register save buffer 13. When saving the pre-update data C' in the register save buffer 13, the content of the counter (II) corresponding to the instruction in the register save buffer counter 20 and the content of the register save address register 12 are counted up by 1. The counter (II) in the register save buffer counter 20 keeps the count number "2" once an update of the software visible register 2 by the instruction B is finished. An update of the software visible register 2 by the following instruction C is conducted by counting up the next register save buffer counter (III) by 1.

Although not indicated in the time chart, in response to an update of the instruction counter 17, data in the register save buffer 13 is cleared, starting from data at the smallest register save buffer address, by the count number held in one of the register save buffer counters (I) ~ (III) corresponding to the update, because the content of the register save buffer 13 becomes unnecessary as a result of the update of the instruction counter. The data in the register save buffer 13 is cleared also in preparation for saving the next instruction and facilitating a recovery operation, which will be described later.

Next, the operation will be explained in detail by way of Fig. 1.

Referring to Fig. 1, when there is no notification of failure detection from the failure processing control means 16 via the line 108, the instruction counter update control means 15 notifies the register save buffer counter control means 19, via the signal line 110, that the instruction was updated. The register save buffer control means 19 instructs the register save buffer counter 20 via the line 119, to notify the register save buffer clear control means 21 via the line 120, of the information in the counter (I), (II) or (III) corresponding to the updated instruction in the register save buffer counter 20.

The instruction counter update control means 15 gives an instruction to clear to the register save buffer clear control means 21 via the line 121. Based on the information in the counter corresponding to the updated instruction provided via the line 120, the register save buffer clear control means 21 clears via the line 122 the validation indication of the data validation bits, starting from the youngest address in the register save buffer 13. The clear operation is conducted by counting down the count number of the corresponding counter in the register save buffer counter 20 by 1 and is continued until the count number becomes "0".

When the failure processing control means 16 is notified that a failure has occurred, recovery processing for starting instruction retry begins in the cycle 11. The addresses indicated in the register safe buffer address register 12 are counted down by 1 at a time and in the cycles 12 ~ 15, the content of the register save buffer is recovered in the software visible register 2, whereby instruction retry is started. An operation of the instruction retry will be described in detail below.

Referring to Figs. 1 ~ 3, recovery processing is started when the failure processing control means 16 sends a recovery instruction to the register save buffer save/recovery control means 11 via the line 111. In response to an instruction provided from the recovery save buffer save/recover control

means 11 via the line 125, all the data B0', B1' and C' whose validity is indicated by the validity bits in the register save buffer 13, type indication signals associated therewith, and register numbers, are stored in the software visible register 2 via the lines 114, 113 and 112 and registers 6, 5 and 4. The validity bits in the register save buffer 13 corresponding to the read data are reset to "0" from "1".

Upon completion of the above-described data recovery to the software visible register 2, the counter number of the register save buffer counter 20 is reset by the instruction to clear provided from the register save buffer save/recovery control means 11 via the line 123. Thereby, the system enters an instruction retry enable state, which makes it possible to redo execution of the instruction from the value kept in the instruction counter since the occurrence of failure.

Although two pipelines, i.e., floating point arithmetic pipeline and fixed point arithmetic pipeline are employed in the above embodiment for the sake of argument, the present invention is in no way restricted by the number of pipelines and can be embodied with three or more operation pipelines.

[Effect of the present invention]

As is described above, according to the present invention, as soon as data is determined, the content of a software visible register is updated and pre-update data is register saved/recovered to enable instruction retry, whereby updated data in the software visible register can be quickly delivered to the next instruction. Thus, the present invention can improve an arithmetic operation speed and achieve a high retry rate with few hardware items.

[Brief explanation of the drawings]

Fig. 1 is an illustration of an embodiment of the present invention.

Fig. 2 is a block diagram indicating a detailed configuration of the operation control means 1.

Fig. 3 is an illustration for explaining an operation of the embodiment of the present invention.

Fig. 4 is a flow chart describing pipeline processing, on which the present invention is based.

Fig. 5 is an illustration for describing prior art operation pipeline processing.

[Description of the referential numerals]

1: operation control means
2: software visible register
3 ~ 10: register
11: register safe buffer save/recovery control means
13: register save buffer
15: register instruction counter update control means
16: failure processing control means
19: register save buffer counter control means
20: register save buffer counter
21: register save buffer clear control means

⑲日本国特許庁（JP）　⑪特許出願公開

⑫公開特許公報（A）　平1－100638

⑤1Int.Cl.⁴	識別記号	庁内整理番号	㊸公開　平成1年(1989)4月18日
G 06 F 11/14	310	N－7368－5B	
9/38	380	A－7361－5B	

審査請求　未請求　発明の数 1　（全9頁）

㊹発明の名称　命令リトライ制御方式

㉑特　願　昭62－259341

㉒出　願　昭62(1987)10月14日

㉒発明者　愛野　茂幸　東京都港区芝5丁目33番1号　日本電気株式会社内

㉒発明者　山野　孝三　東京都港区芝5丁目33番1号　日本電気株式会社内

㉑出願人　日本電気株式会社　東京都港区芝5丁目33番1号

㉔代理人　弁理士　柳川　信

明　細　書

1．発明の名称

命令リトライ制御方式

2．特許請求の範囲

パイプラインの長さの異なる演算制御手段を有する命令リトライ制御方式であって、前記演算制御手段からの演算結果を格納するソフトウェアビジブル格納手段と、このソフトウェアビジブル格納手段の内容がある命令に対していくつ更新されたかを示す値を表示するレジスタセーブバッファカウンタ手段と、どの命令がいくつ前記ソフトウェアビジブル格納手段の内容を更新し前記レジスタセーブバッファカウンタ手段のどの値が表示されているかを認識し制御する手段とを含むことを特徴とする命令リトライ制御方式。

3．発明の詳細な説明

技術分野

本発明はリトライ制御方式に関し、特に間欠障害の救済を行うための命令リトライ制御方式に関する。

従来技術

従来のパイプライン処理型情報処理装置でのパイプライン処理は次のようにして行われる。第4図を参照すると、このパイプライン処理の一例では、アドレス手段により命令キャッシュから命令を取出す命令取出（IF）ステージ、このステージで取出された命令を命令レジスタに格納したあとこの命令のオペランドにもとづきアドレス加算器で論理アドレスを生成するオペランドアドレス生成（AC）ステージ、このステージで生成された論理アドレスを論理アドレスレジスタに格納した後アドレス変換バッファで論理アドレスを物理アドレスに変換するアドレス変換（AT）ステージ、このステージで変換された物理アドレスを物理アドレスレジスタに格納したあと、この物理アドレスでオペランドキャッシュをアクセスし、オペランドを読出すオペランドキャッシュアクセス（CA）ステージ、このステージで読出されたオペランドを実行レジスタに格納したあと演算器で演算する

演算実行（EX）ステージ、及びこのステージで演算された結果を格納する結果格納（ST）ステージの６つのステージに分割されている。

この各ステージは一般には１マシンサイクルタイムｔが割当てられ、この入力端から１マシンサイクルｔ毎にデータが与えられ夫々のステージで論理処理が行われたあと、出力端に１マシンサイクルタイムｔ毎に格納される。このうち演算実行（EX）ステージはさらに数段階の演算実行ステージに分割されており、これは１マシンサイクルタイムｔ以上の時間を必要とする。

第５図を参照すると、演算器の左側には浮動小数点の演算を行うパイプラインステージが構成され、右側には固定小数点の演算を行うパイプラインステージが構成されている。このような構成では、固定小数点演算命令のように右側の一段目のステージで演算が終了してしまうにもかかわらず、左側のソフトウェアステージに合せた固定レジスタが配置されタイミング調整が行われるため、命令カウンタの更新は長い左側の浮動小数点用パイ

プラインによる命令カウンタの更新時まで待たれることになる。従って、固定レジスタを何段も重ねる必要があり、またその制御回路も必要となってハードウェアの増大を招くことになる。その上、演算結果は固定レジスタを順次流れることになり、後続の命令のこの演算実行ステージへの導入が遅延されるという欠点も招く。

発明の目的

本発明の目的は、後続の命令に早く更新されたソフトビジブルレジスタのデータを渡すことにより、演算速度の向上をはかり、少ないハードウェアで高いリトライ率を得るようにした命令リトライ制御方式を提供することにある。

発明の構成

本発明の方式は、パイプラインの長さの異なる演算制御手段を有する命令リトライ制御方式であって、前記演算制御手段からの演算結果を格納するソフトウェアビジブル格納手段と、このソフトウェアビジブル格納手段の内容がある命令に対していくつ更新されたかを示す値を表示するレジス

タセーブバッファカウンタ手段と、どの命令がいくつ前記ソフトウェアビジブル格納手段の内容を更新し前記レジスタセーブカウンタ手段のどの値が表示されているかを認識し制御する手段とを含む構成である。

実施例

次に本発明の一実施例について図面を参照して詳細に説明する。

第１図を参照すると、本発明の一実施例は第４図の命令取出（IF）、オペランドアドレス生成（AC）、アドレス変換（AT）、及びオペランドキャッシュアクセス（CA）の各処理のあと演算実行（EX）の制御を行う演算制御部１、ソフトウェアが操作可能なソフトウェアビジブルレジスタ２、演算制御部１から線101を介して与えられるレジスタ２用更新指示信号を格納する第１更新指示レジスタPCC ３、演算制御部１から線102を介して与えられるレジスタ２用種別指示信号を格納する第１種別指示レジスタRID ４、演算制御部１から線103を介して与えられるレジスタ２のレジスタ

番号を格納する第１番号レジスタRNO ５、演算制御部１から線104を介して与えられるレジスタ２用更新データを格納する更新データレジスタRDR ６、第１更新指示レジスタPCC ３からの更新指示信号を格納する第２更新指示レジスタPCC'７、第１種別指示レジスタRID ４からの種別指示信号を格納するRID'８、第１番号レジスタRNO ５からのレジスタ番号を格納する第２番号レジスタRNO'９、ソフトウェアビジブルレジスタ２からの書込前データを格納する読出データレジスタRDR'１０、ソフトウェアビジブルレジスタ２の更新に応答して更新前のデータをレジスタセーブバッファ１３に書込む指示をし、リカバーに応答してレジスタセーブバッファ１３内の有効データをすべてソフトウェアビジブルレジスタ２に書戻すように指示するレジスタセーブバッファセーブ／リカバー制御部１１、この制御部１１の指示に応答して更新前のデータを記憶し有効データを読出すレジスタセーブハッファ１３、このレジスタセーブバッファ１３に対しクリア指示を行うレジスタセーブバッ

ファクリア制御部２１、線115 を介して前記レジスタセーブバッファ１３にセーブアドレスを供給するセーブアドレスレジスタSBA １２、このレジスタSBA １２の内容を更新するためのカウンタCNT １４、命令カウンタ１７、この命令カウンタ１７の内容を更新するためのCNT １８、障害処理を制御し線108 を介して障害検出信号を送出し線111 を介してセーブ／リカバー制御部１１にリカバー指示信号を送出する障害処理制御部１６、線108 を介して与えられる障害検出信号に応答して命令カウンタ１７の内容の更新を制御する命令カウンタ更新制御部１５、ある命令がいくつのソフトウェアビジブルレジスタ２の内容を更新したかを示す値を保持するレジスタセーブバッファカウンタ２０、及び演算部における複数の段数の異なるパイプラインによりソフトウェアビジブルレジスタ２の更新順序は演算部に入るまでのシリアルな命令順序とは一致しなくなるために、どの命令がいくつソフトウェアビジブルレジスタ２の値を更新しその値であるカウント数をレジスタセーブバ

ッファカウンタ２０の中のどのカウンタ（Ⅰ）、（Ⅱ）または（Ⅲ）が保持しているかを認識し制御するレジスタバッファカウンタ制御部１９を含む。

第２図を参照すると、第１図の演算制御部１は５段のパイプラインで形成されており、浮動小数点演算にともないソフトウェアビジブルレジスタ２用更新指示信号を夫々のステージ（段）で保持するレジスタ群３１～３６、浮動小数点演算にともない更新すべきソフトウェアビジブルレジスタ２種別信号をパイプラインの各ステージで保持するレジスタ群３７～４２、浮動小数点演算にともない更新すべきソフトウェアビジブルレジスタ２のレジスタ番号をパイプラインの各ステージで保持するレジスタ群４３～４８、浮動小数点演算を行うためのレジスタ群４９～５４と、桁合せ回路５５及び５９と、演算器群５６～５８、固定小数点演算にともないソフトウェアビジブルレジスタ２用更新指示信号をパイプラインの最初のステージのみで保持するレジスタ６０及び６１、固定小

数点演算にともない更新すべきソフトウェアビジブルレジスタ２種別信号をパイプラインの最初のステージのみで保持するレジスタ６２及び６３、固定小数点演算にともない更新すべきソフトウェアビジブルレジスタのレジスタ番号をパイプラインの最初のステージのみで保持するレジスタ６４及び６５、固定小数点演算を行うためのレジスタ６６及び６７と演算器６８、浮動小数点用レジスタ３６及び固定小数点用レジスタ６１のどちらか一方からの更新指示信号を選択して線101 に送出するセレクタ６９、浮動小数点用レジスタ４２及び固定小数点用レジスタ６３のどちらか一方からの種別信号を選択して線102 に送出するセレクタ７０、浮動小数点用レジスタ４８及び固定小数点用レジスタ６５のどちらか一方からのレジスタ番号を選択して線103 に送出するセレクタ７１、及び浮動小数点用レジスタ５４及び固定小数点用レジスタ６７のどちらか一方からの演算結果を選択して線104 に送出するセレクタ７２を含む。

まず、本発明の一実施例の前提となる動作について詳細に説明する。まず、先行する命令Ａは浮動小数点演算命令のように長いパイプライン演算処理を必要とするものとする。

第３図を参照すると、サイクル１で命令Ａが命令取出（IF）ステージの処理により取出される。次にサイクル２では、命令Ａはアドレス生成（AC）ステージの処理によりアドレス生成されるとともに、固定小数点演算命令のような短いパイプライン演算ですむ命令Ｂは、命令取出（IF）ステージの処理により取出される。以下同様に、命令Ａはアドレス変換（AT）、及びキャッシュアクセス（CA）の処理がなされたあと、サイクル５から浮動小数点の演算が行われる。命令Ｂはアドレス生成（AC）、アドレス変換（AT）及びキュッシュアクセス（CA）の処理がなされたあと、サイクル６から３２ビット固定小数点データの１６バイトずつの演算が行われる。

命令Ｃに関しては、命令Ｂに続いてサイクル３から命令取出（IF）、アドレス生成（AC）、アドレス変換（AT）及びキャッシュアクセス（CA）の

処理がなされたあと、サイクル８から１バイト固定小数点データの演算が行われる。

第１図、第２図及び第３図を参照すると、命令Ａの演算はサイクル５で桁合せ回路５５により桁合せ処理がなされたあと、サイクル６，７及び８で演算器群５６，５７及び５８により浮動小数点演算が行われる。

一方、命令Ｂのオペランドはサイクル５で取出されたあと、前半の１６ビットのデータが次のサイクル６で固定小数点演算器６８により演算される。次に後半の１６ビットのデータがサイクル７で該演算器６８により演算される。このようにして演算された結果はサイクル７及び８でレジスタ５４、セレクタ７２を介してデータレジスタ６にセットされる。

次にソフトウェアビジブルレジスタ２からの読出及び書込動作について詳細に説明する。

第１図及び第３図を参照すると、サイクル７で更新データレジスタ６への演算結果のセットとともに、更新指示レジスタPCC ３に更新指示信号が

及びレジスタ９からのレジスタ番号がレジスタセーブバッファ１３に格納される。

一方、ソフトウェアビジブルレジスタ２にはレジスタ６からの演算結果が格納される。このサイクル８では、セーブ／リカバー制御部１１から線118 を介してレジスタセーブバッファカウンタ制御部１９から起動され、制御部１９内の疑似命令カウンタ（IC'）には命令Ｂが設定されるとともに、制御部１９は信号線117 を介して初期設定されていたレジスタバッファカウンタ２０の応答するカウンタⅡをカウントさせる。

演算制御部１から線107 を介して命令カウンタ更新制御部１５が起動される。命令カウンタ更新制御部１５は線108 を介して送られてくる信号で障害検出のないことを判定し、線107 を介して与えられてくる信号により演算制御部１で実行されている命令Ｂが当初予定していた命令Ａでないことを判定し、命令カウンタ１７ではなく、レジスタバッファカウンタ制御部１９）にある疑似命令カウンタに命令Ｂがカウントされる。

セットされ、第１種別指示レジスタRID ４に種別指示信号がセットされ、第１番号レジスタRNO ５にレジスタ番号がセットされる。この更新指示信号は、レジスタ７及び線105 を介してレジスタバッファセーブ／リカバー制御部１１に与えられ線106 を介してセーブバッファアドレスレジスタ１２を初期化する。このレジスタ１２のアドレスはサイクル毎に＋１ずつカウンタCNT １４によりカウントアップされる。従って、次のサイクル８では、レジスタ３からの更新指示信号に応答して、レジスタ４からの種別指示信号及びレジスタRNO ５からのレジスタ番号にもとづきソフトウェアビジブルレジスタ２から更新前のデータBO' を読出し、レジスタ１０を介してレジスタセーブバッファ１３に格納する。

この時有効性ビットは“１”とされ、セーブ／リカバー制御部１１から線116 を介してライト指示信号が与えられるとともに、アドレスレジスタ１２から線115 を介してアドレス“１”が与えられる。このとき、レジスタ８からの種別指示信号

サイクル９はサイクル８と同じ動作が次の命令またはデータに対して行われる。

本発明の特徴は、このステージEX５で演算実行中に障害が検出された場合の命令リトライにある。すなわち、障害検出は障害処理制御部１６は線108 を介して命令カウンタ更新制御部１５に通知される。サイクル１０でこの制御部１５は制御部１９内の疑似命令カウンタをリセットするとともに、線109 を介して命令Ａをセットする。

サイクル１０では、ソフトウェアビジブルレジスタ２から更新前のデータＣ´がレジスタセーブハッファ１３にセーブされる。レジスタセーブバッファ１３へのセーブでは、命令に対応するレジスタセーブバッファカウンタ２０の対応するカウンタ（Ⅱ）の内容とレジスタセーブアドレスレジスタ１２の内容を＋１カウントアップする。レジスタセーブバッファカウンタ２０内のカウンタ（Ⅱ）は命令Ｂによるソフトウェアビジブルレジスタ２の更新が終了するとカウント数“２”を確保し続ける。後続の命令Ｃによるソフトウェアビ

ジブルレジスタ2の更新は、次のレジスタセーブバッファカウンタ（Ⅲ）を＋1ずつカウントアップしながら行われる。

なお、タイムチャートでは、図示されていないが、命令カウンタ17の更新に応答して、それに対応するレジスタセーブバッファカウンタ（Ⅰ）～（Ⅲ）のうちのいずれかの保持カウント数だけ、レジスタセーブバッファアドレスの小さいものからレジスタセーブバッファ13内のデータがクリアされる。これは命令カウンタの更新により、レジスタセーブバッファ13の内容が不要になるためであり、また次の命令のセーブに備えるためであり、その上後述するリカバー動作を容易にするためでもある。

この動作を第1図を参照して詳細に説明する。第1図を参照すると、線108を介して障害処理制御部16から障害検出通知がない場合、命令カウンタ更新制御部15は信号線110を介してレジスタセーブバッファカウンタ制御部19に命令の更新が行われたことを通知する。レジスタセーブバ

ッファカウンタ制御部19は線119を介してレジスタセーブバッファカウンタ20内の更新された命令に対応するカウンタの情報を線120を介してレジスタセーブバッファクリア制御部21に通知するよう指示する。

命令カウンタ更新制御部15はレジスタセーブバッファクリア制御部21に線121を介してクリア指示を行う。レジスタセーブバッファクリア制御部21は線120を介して与えられる更新された命令に対応するカウンタの情報をもとに、線122を介して、レジスタセーブバッファ13の若いアドレスからデータ有効性ビットの有効表示をクリアする。このクリア動作は、レジスタセーブバッファカウンタ20内の対応するカウンタのカウント数を－1カウントダウンしながら行われ、カウント数が“0”になるまで繰返される。

障害発生が障害処理部へ報告されると、命令リトライを開始するためのリカバー処理が、サイクル11から行われる。レジスタセーフバッファアドレスレジスタ12で示されたアドレスから－1

ずつカウントダウンされ、サイクル12～15において、レジスタセーブバッファの内容がソフトウェアビジブルレジスタ2にリカバーされ、命令リトライが開始される。この動作を以下詳述する。

第1図及び第3図を参照すると、リカバー処理は、障害処理制御部16が線111を介してレジスタセーブバッファセーブ／リカバー制御部11にリカバー指示を出すことにより開始される。このリカバーセーブバッファセーブ／リカバー制御部11から線125を介して与えられる指示に応答して、サイクル12，13及び14においてレジスタセーブバッファ13内の有効性ビットにより有効性表示されている全てのデータB0′，B1′及びC′、更にこれに付随する種別指示信号及びレジスタ番号が線114，113及び112、レジスタ6，5及び4を介してソフトウェアビジブルレジスタ2に格納される。これとともに読出されたデータに対応するレジスタセーブバッファ13内の有効性ビットを“1”から“0”に戻す。

このソフトウェアビジブルレジスタ2へのデータリカバー完了後、レジスタセーブバッファセーブ／リカバー制御部11から線123を介して与えられるクリア指示によりレジスタセーブバッファカウンタ20のカウント数がリセットされる。これにより、命令リトライ可能状態となり、障害発生時点から保持し続けている命令カウンタの値から命令をやり直すことができる。

この実施例では説明の便宜上、パイプラインを浮動小数点用演算パイプラインと固定小数点用演算パイプラインとの2本で説明したが、本発明はこれに限定されず3以上の演算パイプラインでも実施可能である。

発明の効果

以上説明したように、本発明によれば、データが決定されるとソフトウェアビジブルレジスタの内容をすぐに更新し、更新前データをレジスタセーブ／リカバーし命令リトライを可能とすることにより、後続の命令に更新されたソフトウェアビジブルレジスタのデータを早く渡してやれるので、演算速度の向上を計り、また少ないハードウェア

量で高いリトライ率を得ることができるという効果がある。

4．図面の簡単な説明

第1図は本発明の一実施例を示す図、第2図は第1図の演算制御部1の詳細な構成例を示す図、第3図は本発明の一実施例の動作を説明するための図、第4図は本発明の前提となるパイプライン処理を説明するための図，第5図は従来の演算パイプライン処理を説明するための図である。

主要部分の符号の説明

1……演算制御部

2……ソフトウェアビジブルレジスタ

3～10……レジスタ

11……レジスタセーフーバッファ

セーブ／リカバー制御部

13……レジスタセーブバッファ

15……レジスタ命令カウンタ更新制御部

16……障害処理制御

19……レジスタセーブバッファカウンタ制御部

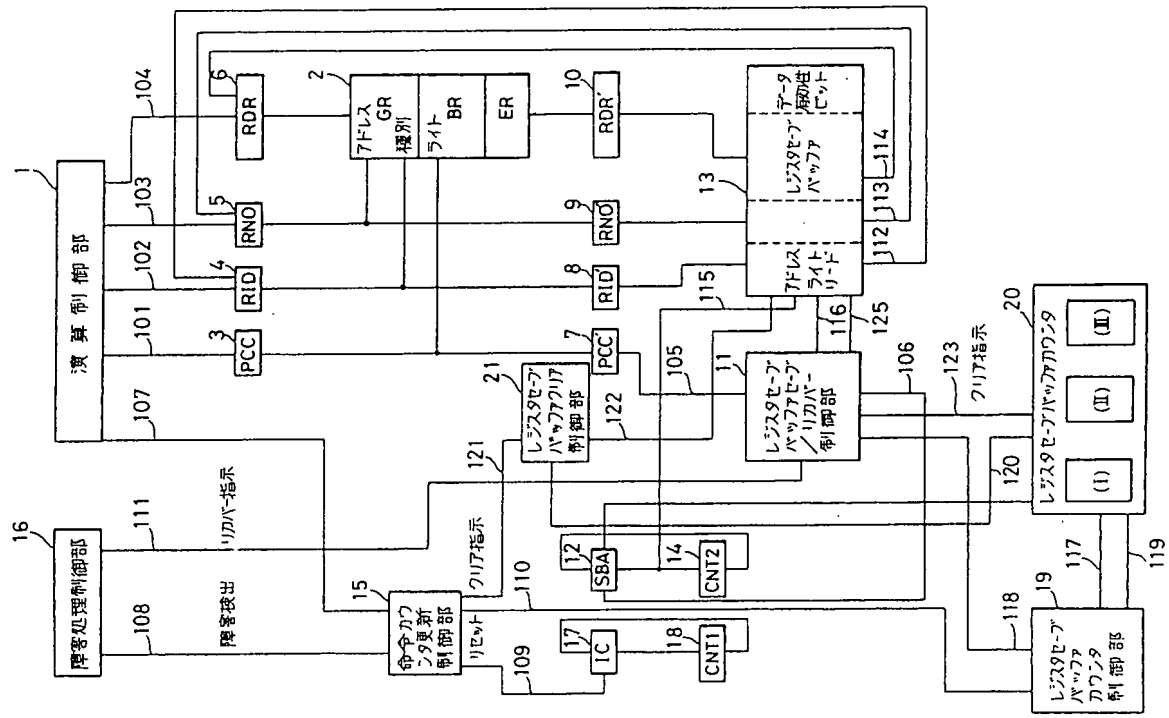
20……レジスタセーブバッファカウンタ

21……レジスタセーブバッファクリア制御部

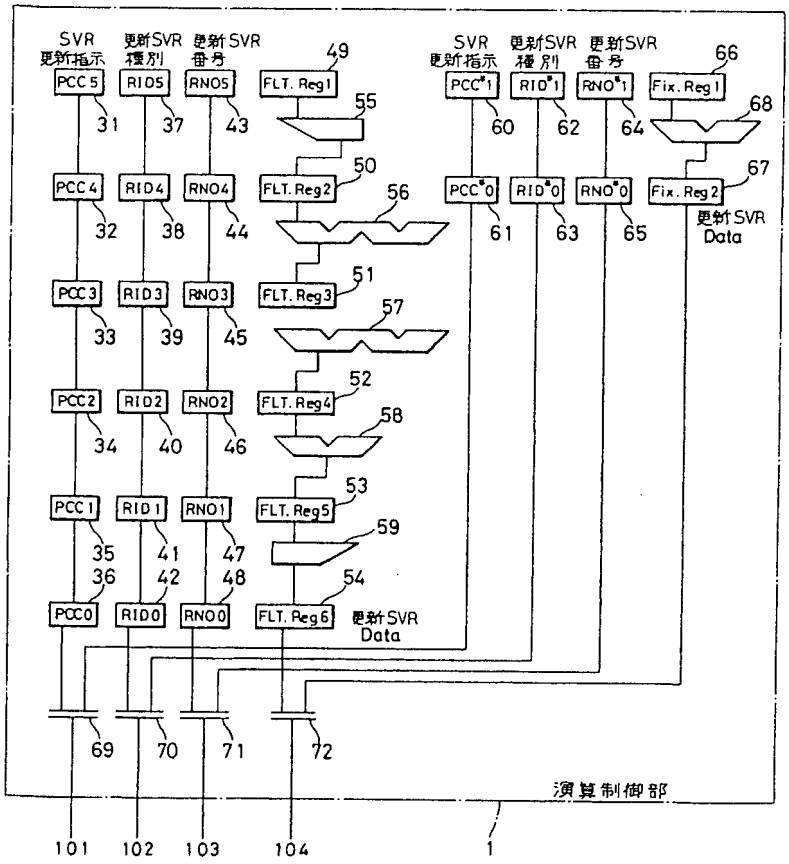
出願人　日本電気株式会社

代理人　弁理士　柳川　信

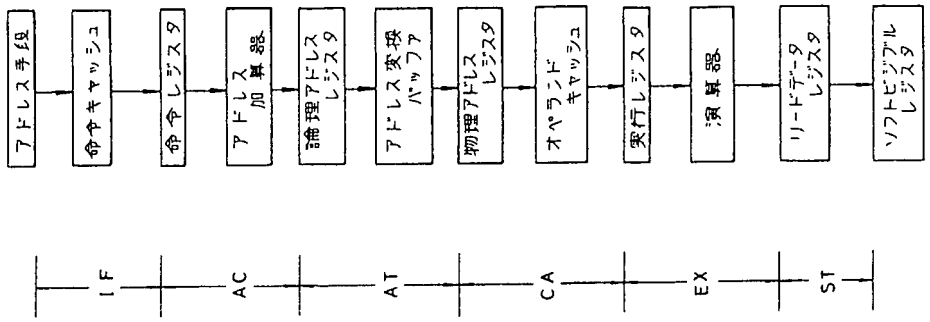
Fig. 1
第1図



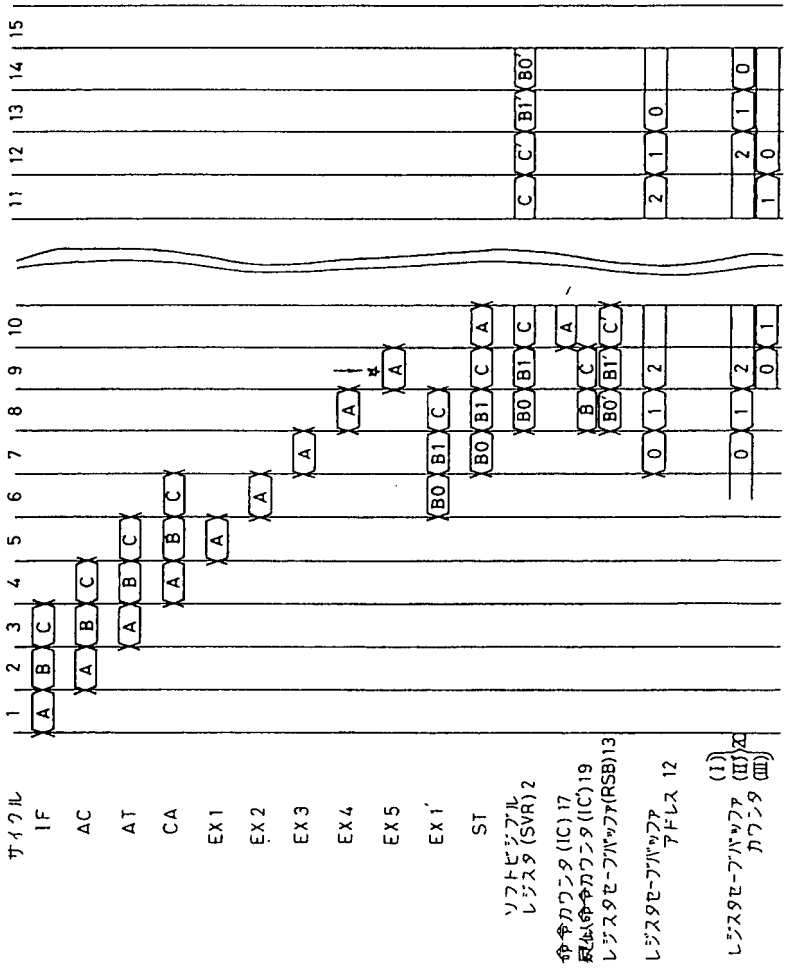
第 2 図


SVR 更新指示
更新SVR 種別
更新SVR 番号
PCC5
RID5
RNO5
FLT. Reg1
49
31
37
43
55
PCC4
RID4
RNO4
FLT. Reg2
50
32
38
44
56
PCC3
RID3
RNO3
FLT. Reg3
51
33
39
45
57
PCC2
RID2
RNO2
FLT. Reg4
52
34
40
46
58
PCC1
RID1
RNO1
FLT. Reg5
53
35
41
47
59
PCC0
RID0
RNO0
FLT. Reg6
54
36
42
48
更新SVR Data
SVR 更新指示
更新SVR 種別
更新SVR 番号
PCC*1
RID*1
RNO*1
Fix. Reg1
66
60
62
64
68
PCC*0
RID*0
RNO*0
Fix. Reg2
67
61
63
65
更新SVR Data
69
70
71
72
演算制御部
1
101
102
103
104

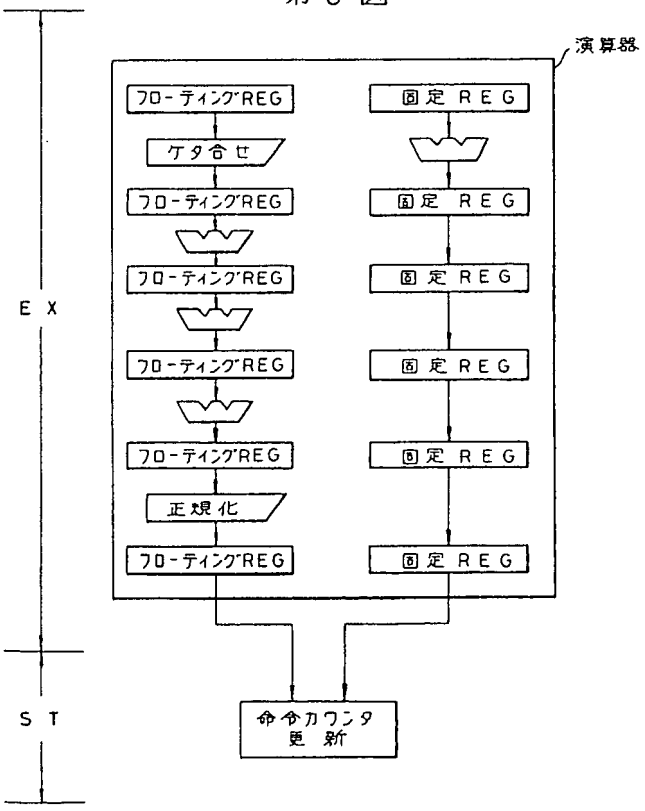
第4図


アドレス手段
命令キャッシュ
命令レジスタ
アドレス加算器
論理アドレスレジスタ
アドレス変換バッファ
物理アドレスレジスタ
オペランドキャッシュ
実行レジスタ
演算器
リードデータレジスタ
ソフトビジブルレジスタ
IF
AC
AT
CA
EX
ST

第3図


サイクル
IF
AC
AT
CA
EX1
EX2
EX3
EX4
EX5
EX'
ST
ソフトビジブルレジスタ (SVR) 2
命令カウンタ (IC) 17
退避命令カウンタ (IC') 19
レジスタセーブバッファ (RSB) 13
レジスタセーブバッファアドレス 12
レジスタセーブバッファカウンタ (I) (II) (III)

第 5 図


演算器
フロ－ティング'REG
ケタ合せ
フロ－ティング'REG
フロ－ティング'REG
フロ－ティング'REG
フロ－ティング'REG
正規化
フロ－ティング'REG
固定 REG
固定 REG
固定 REG
固定 REG
固定 REG
固定 REG
E X
S T
命令カウンタ
更 新